SDCalc Design

Robert Garwood
Harvey Liszt
Paul Shannon
National Radio Astronomy Observatory
Charlottesville, Virginia, U.S.A.

Text last updated: 28 October 1996
HTML version available at:
`http://www.cv.nrao.edu/~bgarwood/sdcalc/sdcalc.html`

Contents

Comments should be sent to aips2-astro-design@nrao.edu.

1 Overview

SDCalc is an aips++ tool for viewing, editing, and operating on one-dimensional data structures typical of single dish telescopes. It is a single dish vector calculator. A previous specification for SDCalc is found in a note prepared by Liszt and turned into html by Garwood[1].

This document sketches out the design of SDCalc. SDCalc is a collection of Glish scripts using a wide range of aips++ distributed objects and Glish clients. A GUI is being developed which uses these Glish commands to present the SDCalc environment to users. The philosophy of the SDCalc user environment is described by Shannon in aips++ note 194[2]. As the actual implementation proceeds it is anticipated that the design will change. This document will be kept up to date.

SDCalc depends on a number of general aips++ developments, however, we do not wish to delay SDCalc if at all possible. As a consequence, we anticipate that we will need to adopt some intermediate shortcut solutions to problems until a more permanent solution is available. This document will describe all such temporary solutions so that they are publicly recognized as such and so that future work to implement long-term solutions can be planned.

2 Architecture

The design of SDCalc from the users point was described initially by Liszt[3]. From the novice and infrequent users perspective, it is vital that SDCalc be seen as a cleanly-integrated application. Experienced users will want to move beyond the default facad which SDCalc presents to the aips++ tools from which it is constructed. In aips++ note 194[4] Shannon presents a model for the SDCalc user environment that is designed to allow users to migrate from the default SDCalc GUI to a direct interaction with the AIPS++ tools that make up SDCalc. Aips++ note 194 describes two tools designed to help users understand the inner workings of the SDCalc GUI: a Transparency Tool and an Evaluation Tool.

The Transparency Tool will return a nicely formatted version of a Glish function's text. Further decomposition may be achieved by calling the tool on a function which appears in the first decomposition. It must be possible to redirect the function listing to the Glish tty, to a Tk edit buffer widget, or to a named file. Users of SDCalc will be able to see the Glish which coresponds to each GUI event if they so choose. By examining the Glish that the standard SDCalc GUI uses along with the opportunity for further decomposition that the transparency tool will provide, users should be able to learn about the tools used to build SDCalc.

The Evaluation Tool is a text editing buffer, probably a Tk text widget, from which text will be sent to the Glish interpreter. We should be able to send selected text, the whole buffer, or the function in which the cursor is currently located.

[1] http://www.cv.nrao.edu/~bgarwood/sdcalc-specs.html
[2] http://aips2.nrao.edu/aips++/docs/notes/194.ps
[3] http://www.cv.nrao.edu/~bgarwood/sdcalc-specs.html#section7
[4] http://aips2.nrao.edu/aips++/docs/notes/194.ps

The SDCalc GUI will use these tools to provide a view into what goes on which each GUI operation. Users will be able to see the the Glish command that each GUI operation uses to do the actual work. These commands can be viewed in the Transparency tool (which can then be used to further decompose the high level Glish which the GUI uses) or copied to the evaluation tool for editing (which could then be turned into a script or function to encapsulate repetative operations that the user finds useful for that particular data set).

2.1 Broad Outline

The table system is the principle data structure for long term storage in aips++. Within this system, a convention known as the MeasurementSet has been adopted to describe all astronomical data. This convention uses an agreed upon set of standard table column names along with an agreed organization of a main table and several subtables to hold astronomical data. It is necessarily a complex structure because, in general, astronomical data is complex. Within aips++, users will typically never interact directly with the MeasurementSet. Instead, specific, limited views of the MeasurementSet that are tailored for certain analysis goals are constructed as needed. Users interact with the MeasurementSet through a specific view.

In the synthesis imaging work in aips++, a VisibilityIterator and a VisSet have been developed. The VisibilityIterator is the view of the MeasurementSet. It determines how the user steps (iterates) through through the MeasurementSet. Specific values common to synthesis imaging needs are available as function calls to the iterator (rather than as columns in the MeasurementSet). The view may in fact present derived quantities not found in the MeasurementSet. The VisSet is an interface to the MeasurementSet that manipulates reference copies of the MeasurementSet. The VisSet contains three VisibilityIterators: observedCoherence, modelCoherence, and calibratedCoherence. Values are placed back into a MeasurementSet through an iterator. In a general sense, these concepts will be used within SDCalc.

SDCalc is designed around an atom called an SDRecord. An SDIterator is the view of the MeasurementSet that SDCalc will use. An SDIterator serves up individual SDRecords or collections of SDRecords to the rest of SDCalc. It may ultimately be desirable to have the single dish equivalent of the VisSet (an SDSet). This might include, for example, processedData and observedData to associate these two iterators more closely.

SDCalc will manipulate and display SDRecords. There will be an "active" SDRecord that SDCalc will display automatically after each operation if the user chooses to do so. Operations will place ther results into that active location when appropriate. A simple "undo" is then possible by keeping a copy of the active record before each operation and swapping that copy when "undo" is requested. This only allows for a single level of undo. More complicated undo schemes will eventually be considered if necessary.

Within Glish, an SDRecord is a Glish record consisting of the data, a header holding standard parameters associated with the data (either at the time of the observations or during the processing within SDCalc), a non-standard header holding all other ancilliary information (e.g. telescope specific information), and a history of the SDCalc operations that have been done to the data. Each of these parts, data, header, non-standard header and history, is in turn a Glish record.

The data record will consist of a two-dimensional array and the necessary fields to describe each axis of the array. The first axis can be any linear axis in whatever coordinate system the user chooses. The second axis indicates the polarization or stokes values. Usually there will be either 1, 2, or 4 pixels along this axis. The first axis will be referred to as the x-axis throughout the remainder of this document. The SDRecord is flexible enough to hold all one-dimensional data as long as the x-axis coordinate is linear with pixel number.

The basic data flow within SDCalc is from some data source (a MeasurementSet, an aips++ image, a single dish FITS file) through an SDIterator and into an SDRecord. Operations are done on this SDRecord, possibly combining with other SDRecords and leaving an SDRecord as a result. For efficiency purposes it appears that it will be necessary to support operations on SDIterator (e.g. average all SDRecords contained in a specific SDIterator to produce a single SDRecord, fit baselines to all SDRecords in an SDIterator to produce a new SDIterator, etc.). SDCalc will provide display tools to view a single SDRecord or "browse" an SDIterator (which is mearly a set of SDRecords).

Data selection is done by generating a new SDIterator from the original SDIterator with the selection criteria applied. If the underlying data source uses the Table system (see comments on IO issues later in this document), then such a selection would produce a new SDIterator which was mearly a reference to the selected rows of the original SDIterator (i.e. no data would need to be copied). It must eventually be possible to select on any field in an SDRecord. Operations on SDIterators (either simply retrieving or storing an SDRecord or operations on whole SDIterators) will be identical when using the original data or selected data.

At some stage, the user will want to store an SDRecord. The SDRecord is sufficiently flexible that certain processing may make it inefficient or even impossible for a general SDRecord to be stored in the current design of the MeasurementSet. Other storage options must be provided. Specifically, it should be possible to store an SDRecord into a table of SDRecords. It may become true that a MeasurementSet will be sufficiently flexible that this is no longer required although we suspect that users will value this ability in any case. Whatever the underlying storage mechanism it should be easy to write an SDRecord or set of SDRecords to a single dish FITS file.

2.2 Specifics

2.2.1 SDRecord

The SDRecord is the fundamental data atom of SDCalc.

Unless otherwise stated, the units of all values in an SDRecord are SI units. Angles are expressed in degrees. Until Measures and coordinates are fully supported in Glish, temporary fields in an SDRecord will be used to hold sufficient information to indicate the coordinate system and reference frame where appropriate. All of the fields described below are required unless stated otherwise.

The value of a field is undefined in the following circumstances:

- The Glish function `is_nan(x)` returns `T` for a floating point value.
- The value is < 0 for an integer field.
- The string is empty.

For boolean fields, all values are considered valid. In addition, all values in the data array should be considered valid. The data array flag field should be used to indicate invalid values in the data array.

Certain operations on SDRecords lead to ambiguities in some fields in the resulting SDRecord. In averaging two SDRecords taken at different times, it is not clear what value should be given to the `time` field and the fields that contain values at the given `time` (e.g. `azimuth`). What value should be given to the `direction` when data from different directions are combined? Rather than simply mark these fields as undefined (which would loose all of this information), one of the input records will be chosen as a template for use during that operation. Typically this will be the first record used in that operation. All values in the resulting record except those directly affected by the operation (e.g. `tsys` and `exposure` in an average) will be unchanged from the values in the template record. The documentation for each operation should describe when this occurs. The `hist` field will show the history of operations.

An SDRecord is a Glish record having the following structure:

data A Glish record which contains the data array and a description of the data array (axis type, values, increments, and units).

- **arr** The data array. The dimensionality of the data array is `(nchan, nstokes)` where `nchan` is the number of channels and `nstokes` is the number of different stokes types found in this data. `nstokes` is either 1, 2 or 4.
- **desc** A Glish record which describes the data array.
  - **ctype, crval, crpix, and cdelt** Follow standard FITS usage. They describe the first axis of the data array. An SDRecord can therefore only contain data where the first axis is linear in some coordinate system.
  - **units** A string describing the units of the data array.
  - **stokes** A vector of strings describing the stokes type of each element of the second axis of the data array. The possible stokes types are:

    I, Q, U, V, RR, RL, LR, LL, XX, XY, YX, YY, RX, RY, LX, LY, XR, XL
    YR, YL, PP, PQ, QP, QQ

    For a typical single dish telescope, raw data will have `stokes = ("XX", "YY") or ("RR", "LL")`.
- **flag** A boolean matrix with the shape as arr. When true, the corresponding value in arr has been flagged as bad and should not be used. When false, the corresponding value in arr is good.
- **weight** A floating point matrix having the same shape as arr.

header A Glish record with the following fixed structure. The bulleted items are mearly separators to make this list easier to read, they are not members of header. The type of each field is indicated at the end of each description. Vector fields are indicated by giving the length of the vector in parenthesis after the field type.

- Identity

  time The mid-point of the observation (UT). Double.

  scan_number The scan ID. Integer.

  object The name of the thing being observed. String.

  direction Where the telescope was pointed. Double(2).

  direction_rate Any slew rate during the observation. Double(2).

- Times

  ut_date The UT date corresponding to time. String.

ut The UT on ut_date corresponding to time. Double.

lst The LST corresponding to time. Double.

exposure The total amount of time actually spent collecting photons (e.g excludes blanking time, etc).

duration The elapsed (clock) time. Float.

- Observer

observer String

project String

- Backend

rest_frequency Double.

resolution The frequency resolution. This need not be the same as the channel spacing.

bandwidth The total bandwidth. Double.

obs_frequency The sky frequency at the reference channel. Double.

tcal The cal temperature. Float(nstokes).

trx The receiver temperature. Float(nstokes).

tsys The system temperature. Float(nstokes).

sigma The theoretical noise per channel. Float(nstokes).

- References

reference_position A reference position on the sky for position switched data. Double(2).

reference_frequency The frequency reference for frequency switched data. Double.

- Telescope

telescope A string identifying the telescope. String.

telescope_position The (x, y, z) position of the telescope in the ITRF (VLBI) coordinate system. Double(3).

telescope_diameter The diameter of the dish. More detailed telescope models will no doubt be useful but the parameters to describe these models are beyond the scope of the header record. If standard components could be identified, they would eventually be placed in the header. Until then, additional parameters to describe the shape of the telescope surface should be placed in ns_header. Float.

telescope_mount The mount type. Chosen from the following values: "alt-az", "equatorial", "X-Y", "orbiting", "bizarre".

azel The azimuth and elevation at time. Double(2).

vcorr The velocity of the telescope with respect to the velocity reference frame at time using the given velocity definition.

- Weather

pressure The atmopheric pressure. Float.

dewpoint The dew point. Float.

tambient The ambient temperature. Float.

wind_dir The wind direction. Float.

wind_speed The wind speed. Float.

- Coordinate system kludges

direction_coord A Glish Record describing the coordinate system for the direction field. It has the following structure.

- **name** The coordinate name. String.
- **equinox** The equinox for the coordinate. Double.

reference_coord A Glish Record describing the reference_position field. It has the following structure.

- **name** The coordinate name. String.
- **equinox** The equinox for the coordinate. Double.
- **is_offset** True if this is an offset position, else this is an absolute position. If this is an offset position then it is offset from the reference_position. Bool.

veldef The velocity definition (e.g. "radio", "optical"). String.

velref The velocity reference frame (e.g. "LSR", "HEL", ,etc.). String.

hist This vector of strings contains a history of the operations which resulted in this SDRecord. Each element is a valid Glish command. Taken as a while, the hist field should be sufficient to regenerate the SDRecord from the original data source.

In addition to the fixed structure described above. An SDRecord may contain an optional `ns_header` field. This is a Glish record which can contain any additional information not found in the fixed portion of an SDRecord. This field will typicall be used for telescope-dependent information.

The x-axis can be easily constructed from the `data` field by the following simple Glish function:

```
make_xaxis := function(ref data)
{
   x := data.crval;
   x +:= ([1:data::Shape[1]] - data.desc.crpix)*data.desc.cdelt;
   return x;
}
```

To summarize, in pseudo-Glish an SDRecord looks like this:

```
sdrecord := [data=[=], header=[=], hist=[=], ns_header=[=]];
sdrecord.data := [arr=[...], flag=[...], desc=[=], weight=[...]];
sdrecord.data.desc := [units=,stokes=,ctype=,crpix=,crval=,cdelt=];
        sdrecord.header := [time=,scan_number=,object=,etc.];
sdrecord.hist := ["# useful history","# as a vector","# etc. etc."];
        sdrecord.ns_header := [non-standard fields]

ns_header is optional
```

Issues A list of unresolved issues related to SDRecord:

- What impact with the planned visibility in Glish of the aips++ Measures, units and coordinate information have on data.desc?
- It should be possible to associate multiple data fields with a single header - this would be particularly useful for closely related data - such as that found in an image. This needs to be described.

- History - SDCalc functions need to add to this. What does it look like? How much bloat will this cause if users forget to trim this record? Which functions should add to this - all functions which add information to an SDRecord or alter existing information or only those which alter existing information?
- The values listed above only cover some of the information which can be found in a MeasurementSet. Is this a sufficient list? There is also some information which is missing from the standard MeasurementSet. At some point this deficiency should be corrected. What changes would be necessary to a MeasurementSet so that it would be easy to store the more flexible data structure of an SDRecord (i.e. that in data.desc)?

2.2.2 SDIterator

An SDIterator iterates through a data source one SDRecord at a time. The SDIterator is the view of the data source that SDCalc will use. Other views (e.g. the full MeasurementSet, a flat table view of the MeasurementSet, etc.) will also be supported. An SDIterator is a set of SDRecords. In the SDCalc GUI an SDIterator is referred to as a "working set" to reflect the fact that at any given instant a specific SDIterator (or set of SDRecords) is the data source with which the user is currently working.

The only type of data source we support now is an AIPS++ table filled directly from a single dish FITS file. The data souce will eventually be a primarily a filled MeasurementSet but it must also must be include the types of data listed in the next section. It may eventually be useful to have an SDSet which would contains a number of related iterators, but this is left for the long term development of SDCalc. Data selection is done when the iterator is constructed. Iterators will largely be invisibly to the SDCalc user. Iterators can be constructed (and selected) from other iterators.

An SDIterator is able to get and put individual parts of an SDRecord (especially the `header` as separate from the `data`) in addition to the whole SDRecord. It is often useful to examine just the header values without having to drag the entire data array around with the header.

2.2.3 The SDCalc GUI

The graphical user interface for SDCalc is a work in progress, and currently caters to the somewhat narrow needs of the first operation it provides: spectrum averaging. Some (possibly most) of the current GUI will appear in the the more mature, and more complete program.

Here are the key features:

1. A menubar provides a standard "File" pulldown menu, with menubuttons "Open" and "Exit"; the Open button leads to a standard file selection dialog box, for finding aips++ tables (or optionally, any kind of file on disk).
2. At the top of the main program window, just below the menubar, there is a "Results Manager" – all opened tables, working sets, selections, and other intermediate results appear here, by name, as selectable items in a scrollable list. This manager also provides some description of the currently-selected result, and allows you to rename the result if you wish. "View" and "Delete" buttons are also provided. Viewing is a data-type-specific operation. For instance,

viewing a table means (more accurately: will soon mean) to run the tablebrowser; viewing a spectrum means to plot it; viewing a working set will probably mean to popup a "working set browser" which allows you to navigate through the scans, view header values, and plot spectra at will.

3. A selection tool is in the center of the main program window. This consists of (currently) six comboboxes – a Windows-inspired GUI component that combines a 1-line text entry and display widget, and a scrollable list widget, with a couple of control buttons. There is a combobox for the following categories of data in an SDRecord header or SDIterator:

   Object The object field.

   Record number The record number in the SDIterator.

   Scan number The scan_number field.

   Date The ut_date field.

   LST The lst field.

   Rest frequency The rest_frequency field.

   All of the unique values in each category are extracted from the current input data source (usually an sdfits file previously converted to an aips++ table, but optionally a subset of such a table, created by sdcalc). Each of these values is then added to the list associated with each combobox, and with a few simple mouse actions you can specify the subset of the data you wish to consider for subsequent operations (currently, just spectrum averaging).

4. At the bottom of the main program window is the "Average Spectra" window, with a few buttons for specifying alignment and weighting options, and another button to actually calculate the average, using the current selection (as specified in the selection tool) with the specified options. Once calculated, the average spectrum appears as a variable in the results manager (see item 2 above); the "view" button plots this average in the plotting window. (Note that the plotter, not previously mentioned, is a separate aips++ program, which runs in its own window, and which communicates over the glish software bus.

2.2.4 IO Issues

Most of the IO issues are discussed elsewhere in this document. Data flows from a data source to an SDRecord through an SDIterator and data flows in the other direction as well, when the data source allows it.

- Single Dish FITS - there are two possible paths here. An SD-FITS file can be converted to and from a MS or an SDIterator can read an SD-FITS file directly. Obviously the first path is required but we believe that users will also want the second path, especially on output. Certainly for sufficiently large data sets the conversion step would be extremely time consuming and there might simply not be enough data for two copies of the entire data set. This requires random access FITS classes which can be used to build a FITS storage manager so that an SD-FIT file can appear as a MS.

- Aips++ Image - certainly all images can be viewed as a collection of SDRecords where the x-axis is any of the axes in the image. It should be possible to do just that. It should also eventually be possible to construct an SDIterator that makes slices through an image at an arbitrary orientation to the axes of the image.

- ASCII output - users will want to write out SDRecords into ASCII files. If they want this as an output format, then they will need to be able to convert these into a MeasurementSet or SD-FITS to be used as an iterator. Since this type of data set is very likely to be small, it is not unreasonable that a conversion be done if this is to be used as a data source.
- UniPOPS SDD format - this is essentially the same issue as SD-FITS. We must support the path through a MeasurementSet but users will appreciate a more direct path. We currently have a storage manager for SDD for the old, flat, MeasurementSet so in principle a MeasurementSet could be constructed from an SDD file.
- For most (all?) of the above, we need to be able to write to an existing file (append only in some cases or replace if possible).

3 Operations

The following summary of operations that SDCalc will provide was generate by a quick re-reading of the UniPOPS manual plus a re-reading of the wish list for the GBT that Rick Fisher has written. These are the immediate operations (for beta release) that SDCalc will offer. Additional operations will be added. A section of this document should eventually appear describing those additional operations.

Typical Calibration Schemes There are several levels here. The GBT needs these schemes in order to do the very basic operation of combining the several sub-integration phases (on, off, on+cal, off+cal, etc.) into a single integration. More generally, there are probably a small set of standard operating modes used universally that we should be able to reduce automatically (position switched, frequency switched, beam switched, etc). The issue here, can these be described in a telescope independent manner. The sub-integration to integration averaging will in general be telescope specific. It would be nice if we could plug in telescope specific methods to some uniform place in SDCalc. The generic calibration operations need to be catalogued and described.

Shift and Align Given a starting SDRecord, shift any other SDRecord with a compatible x-axis so that the x-axes align. There should be a number of user controllable shift methods from integer only, to simple spline interpolation to a full FFT based shift.

Average SDRecords together Allow for different averaging schemes.

Fitting The result should go in a number of different places as indicated by the user (attach as a parameter vector or fitted values to the SDRecord, to a Glish record, use it immediately by removing it from the data, store the residuals in place of the data). For baseline removal, it should be possible to use the baseline from another SDRecord (or even use an SDRecord as if it were a fitted baseline, which it might in fact be).

- polynomial
- multiple Gaussians with full control over which parameters are fixed in absolute or relative position.
- multiple sinusoid
- spline
- combinations of functions (e.g. polynomial plus Gaussian)

Smoothing Boxcar, Hanning, Gaussian, arbitrary function

- keeping the same x-axis values
- replacing with new x-axis values appropriate to the new resolution and appropriate re-sampling of the smoothed data.

Interpolation To a different x-axis.

General FFT tools
- FFT and inverse
- Cross-Correlation
- Auto-Correlation
- Convolution

Moments

Galaxy profile analysis Stolen from UniPOPS which in turn was stolen from Analyze. This is included here for completeness. The operations summarized here represent all of the one-dimensional functionality of UniPOPS not already found in Glish. This capability will be developed last after a few rounds of feedback from the initial users of SDCalc.

4 Other AIPS++ Dependencies

SDCalc is relying on ongoing development within aips++. The following are known items SDCalc will need to pay attention to and use when ready.

Help A prototype help system written in will be available sometime this fall.. We will use this as soon as it is ready.

Parameters and state We need to be able to store previously used parameters and other state information. It is not clear where this stands within aips++. We will need to use a short term solution.

MeasurementSet development The MeasurementSet has just been reviewed. The primary data axis remains frequency. The design of SDCalc is intentionally isolated from that of MeasurementSet however the SDIterator will need attention whenever MeasurementSet changes. It will continue to be impossible to store a general SDRecord in a MS.

Single dish FITS IO An initial SD-FITS reader and writer to and from a MeasurementSet is still not available. This will eventually be the main path of UniPOPS DATA into aips++ and hence SDCalc. A direct FITS storage manager is required so that an SD-FITS file can be used as an SDIterator.

UniPOPS SDD IO In order for the 12m and 140' to be useful beta sites it must be easy to get SDD data into SDCalc. This can initially be via SD-FITS (which will require some work in UniPOPS by Garwood) but eventually users will expect a MeasurementSet to be built directly from an SDD file (i.e. the SDD storage manager will need to be revived). This is not required for a considerable length of time.

M&C While not an aips++ dependency, the activity of GBT Monitor and Control will impact SDCalc development. This includes the long term support we have been providing for engineering work. It also includes dealing with new backends and new Data Associated Parameter files and changes in the existing DAP files. This is difficult to predict.

5 Development Plans

5.1 Prototype SDCalc - an averager.

Because it is a basic activity which tests the I/O subsystem and brings together a wide range of collateral considerations, we have created a working prototype of SDCalc's averaging of spectra, along with its GUI. The GUI includes a mechanism for data selection as well as controls for how that averaging is to be done. We are currently refining this prototype so that it does what we think it should do. Later this fall, we will use this prototype to generate some feedback from internal users on the GUI aspect. As the averager becomes more stable and reliable we will gradually add additional features from the above list. The next type of operations we will consider is fitting - specifically fitting a baseline to the data.

5.2 User interface issues

The following questions remain unanswered.

- How does the user ask to see an SDRecord?
- How does she know what data she has?
- How does he interact with or extract information from what he sees?
- How does SDCalc present a summary of an SDRecord's header?

5.3 AIPS++ and GBT dependencies

This is a list of aspects of on-going aips++ development SDCalc requires or that may affect SDCalc design plans plus GBT plans that will affect SDCalc development.

- SDCalc requires the following from AIPS++
  - Glish changes required by desired user environment (transparency issues)
  - Measures/coordinates/units in Glish
  - Help system
- The following planned AIPS++ development will impact SDCalc
  - MeasurementSet review/revisions - in progress.
  - Planned DO revisions - in progress.
- The following GBT plans will affect SDCalc development
  - GBT Holography backend added to filler
  - M&C refinement of contents of FITS files
  - GBT Holography tests in 1997.